Time Animation Report

Chris Sinclair
January 13th, 2024

This report serves as a breakdown of the data that I tracked for my major animation project in 2023. The project itself was an entry for an animated music video competition hosted by the band, Pink Floyd, to celebrate the 50th anniversary of their penultimate 1973 album, “The Dark Side of the Moon”. Having only animated as a hobby in the past, this was my first time putting together a working spreadsheet to track hours worked per day throughout the year (starting on April 1st, 2023). I also divided the spreadsheet up into separate columns to identify each particular section of the song that I was working on animation for – the columns being “Intro”, “Verse 1”, “Guitar Solo 1”, “Guitar Solo 2”, “Verse 2”, and “Outro”.

After finishing the entire animation and spreadsheet, some of the major data points that stick out are listed below. These are all fairly simple statistics that don’t require extensive code, so I’m just going to list them below without a breakdown of the process:

- Total hours worked (2023): 676
- Average Hours per Month: 75.111
- Month with Highest Hours Worked: 120.5 (August)
- Month with Lowest Hours Worked: 44 (October)
- Estimated Hours of Previous Work: 224

For the sake of full disclosure, the “estimated hours of previous work” is, at best, a guesstimate for the amount of hours that were spent on work that ended up in the final video that I had already completed. Most of this work ended up being applied to the “Guitar Solo” section (200 hours), as most of the work was abstract in nature. However, some of it was divided into the “Intro” total (10 hours) as well as the “Verse 2” total (14 hours), based on how much time I think each of those portions took. I also rounded it off for an overall total of 900 hours, mostly due to personal preference. I wouldn’t want the estimated total to be some random number like 886, when I really just don’t know. 900 just looks cleaner while the estimate still seems accurate.

All of the code was done in Python using the Matplotlib, NumPy, Pandas, and SciPy libraries.

Diving into the first specific piece of data that I was interested in, I wanted to find out the average amount of hours worked for each day of the week. The main piece of code that was used for gathering this data is below:

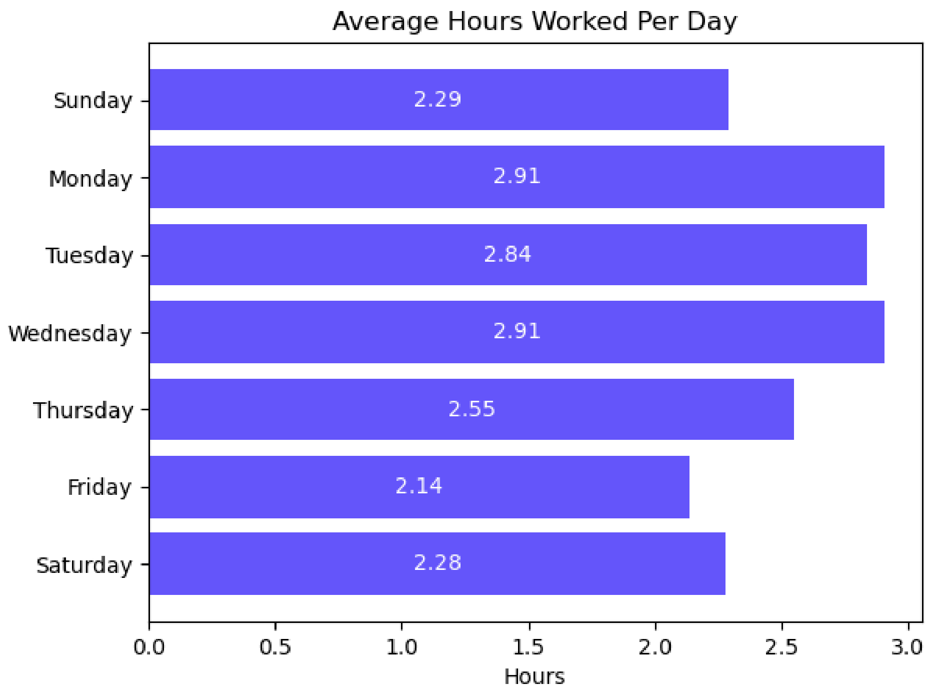
```
sunday_subset = pd[pd["day"] == "sunday"]
print(sunday_subset.count())
```

```
sunday_subset_intro_total = sunday_subset["intro"].sum()
sunday_subset_verse1_total = sunday_subset["verse1"].sum()
sunday_subset_solo1_total = sunday_subset["solo1"].sum()
sunday_subset_solo2_total = sunday_subset["solo2"].sum()
sunday_subset_verse2_total = sunday_subset["verse2"].sum()
sunday_subset_outro_total = sunday_subset["outro"].sum()
sunday_subset_compositing_total = sunday_subset["compositing"].sum()

sunday_total = sunday_subset_intro_total + sunday_subset_verse1_total +
sunday_subset_solo1_total + sunday_subset_solo2_total +
sunday_subset_verse2_total + sunday_subset_outro_total +
sunday_subset_compositing_total

sunday_mean = round((sunday_total / 38), 2)
```

In the spreadsheet, I had a column ("Day") that listed each particular day. I simply had to gather the sum of hours worked from each animated section for each particular day. The code above applies to Sunday, but I ran this code for each day of the week. Once I had a total sum of hours worked for each day, I had to calculate a mean. Given the fact that I began the project on a Saturday but finished on a Wednesday, I knew there would be some days that had more instances than others. I printed a count of how many instances of each day there were (either 37 or 38) and input that number as the denominator to calculate the mean, rounded to two decimal places. Below is the chart with the average amount of hours worked for each day of the week:



Of all the data points, this was the one I was most interested in and actively avoided looking into ahead of time so that I could be sure I'd learn how to do it in developing this report. As seen above, Monday and Wednesday both tied for the most productive days with an average of 2.91 hours each, followed closely by Tuesday with an average of 2.84 hours. I found this surprising because I thought with all the available time I worked on the weekends (sometimes upwards of 6 hours), that Friday, Saturday, or Sunday might come up as a higher number. However, there were plenty of weekends where I simply didn't do much because I was either high on weed, playing video games, or hanging out with friends.

The next chart I was interested in putting together was an overall look at how active I was on the project throughout the year. In the spreadsheet, I had another column called "Week", which simply had an integer detailing which week I was working on, starting from Sunday and ending on Saturday. For example, Sunday, April 2nd to Saturday, April 9th was listed in the "Week" column as 1. And then the next week was listed as 2, and then 3, and so on. With this in mind, I created a new column called "Total" and grouped the total amount of hours worked by each week.

```
pd_line = pd
pd_line["total"] = pd_line["intro"] + pd_line["verse1"] +
pd_line["solo1"] + pd_line["solo2"] + pd_line["verse2"] +
pd_line["outro"]

pd_week = pd_line
pd_week

df_weektotal = pd_week

for week in pd_week:
       pd_week["week total"] =
pd_week["total"].groupby(pd_week["week"]).sum()

df_weektotal = df_weektotal["week total"].dropna()
df_weektotal
```

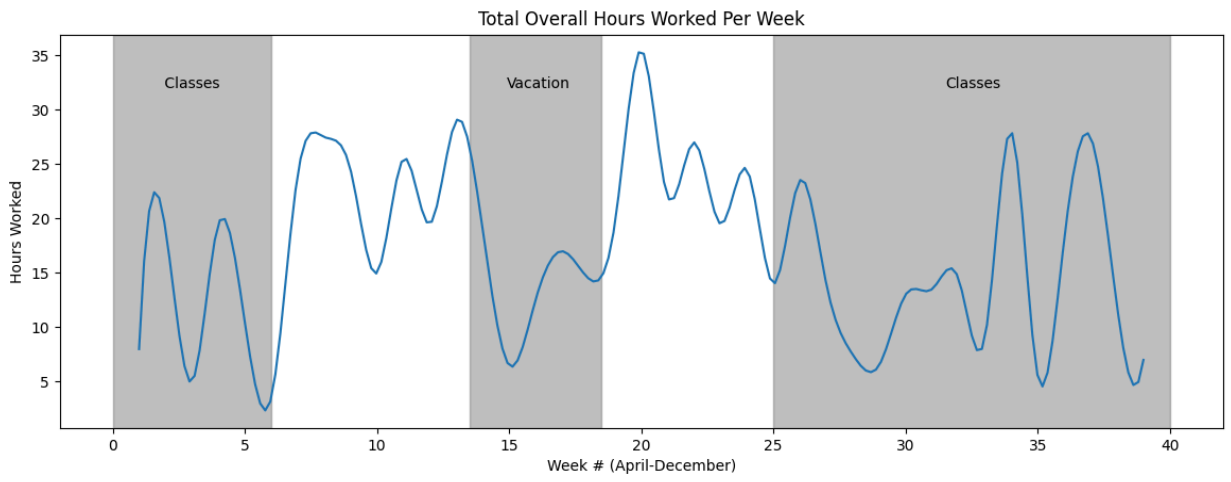
One issue that I ran into here was that when creating these weekly totals in the new "Total" column, it would list the weekly total in each subsequent row rather than the first row of its corresponding week. This created a mismatch where I had the first seven weekly totals listed in the same rows as the first seven days. After failing to correct this issue with the code, and not wanting to manipulate the raw spreadsheet, I decided on a different approach: I would instead isolate the "Week Total" column, and list each row as its own week. From there, I would manually create two lists to serve as x and y coordinates in a line chart. The x list was simply a range from 1-40 to indicate the week, and the y list was an empty array that I appended each weekly total into. Once these lists were created, I turned them into NumPy arrays and was finally able to plot them out.

```
x = list(range(1, 40))
y = []

for i in df_weektotal:
      y.append(i)

arr_x = np.array(x)
arr_y = np.array(y)
```

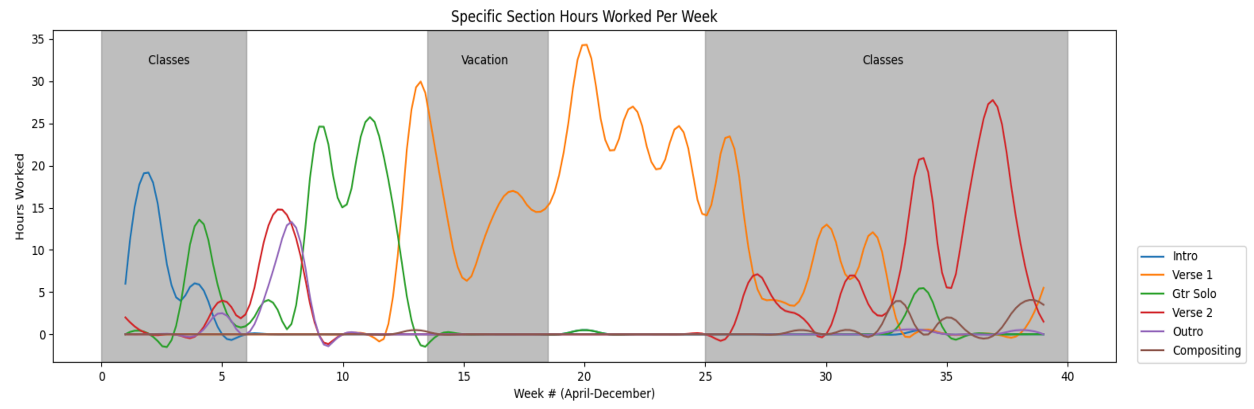
Once plotted, I noticed that a simple line chart was very jagged and hard to interpret, so I imported the SciPy library and looked up how to curve the lines a bit more so the viewer would see a picture of a smoother overall trend in the data points. Below is the breakdown of the total hours worked per week:



Since this project coincided with classes and vacation, I decided to indicate where those specific items came into play so that it would be more clear how that affected the overall weekly trends. As you can see, once classes finished up in early May, I was able to work very intensely through June, with several weeks consisting of over 25 hours of work. However, in July I had two vacations planned and there was a noticeable dip in how many hours were worked per week.

Once the vacation ended, August kicked in and I had an all-time high of 35 hours worked. I'd like to point out that since these weekly totals only ranged from Sunday-Saturday, that specific stretch was actually part of a 40-hour peak from Tuesday-Monday. That week was quite literally the equivalent of working two full-time jobs. Anyways, the amount of hours worked took a slow dip starting in mid-September as my classwork became more demanding, resulting in the month with the least amount of hours worked – October with 44. After that, there are two peaks surrounding a deep valley – that deep valley is Thanksgiving. Once I returned from Thanksgiving break, I worked intensely for the remaining three weeks to finish up the project.

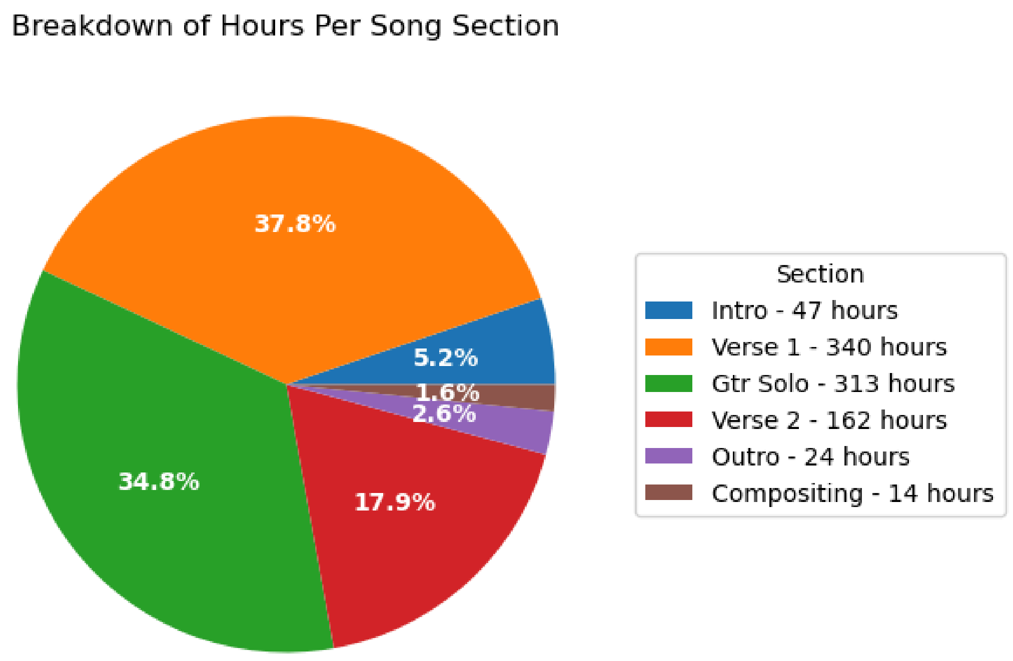
As a follow-up to this chart, I decided to take the same concept and apply it to each individual section of the song to see at what point I was working on which specific section. Below is a breakdown of what I was specifically working on throughout the year:



The three major line colors that stand out are green, orange, and red, which represent the guitar solo, verse 1, and verse 2, respectively. I found it interesting that at some points, I seemed to be working on multiple sections at the same time. This is most noticeable between weeks 6-8, as well as weeks 30-35. Even though the individual peaks are smaller than the main chart above, you can see how the mixed hours on this chart compound to create the taller peaks on the other chart.

I will note that, although the concept of gathering this info was relatively easy, the code is rather messy. For anybody analyzing the code of this particular section, I know that there would have been a much easier way to gather all of this information more efficiently – I just decided to copy and paste the code used to generate the above chart for what I thought would be the sake of simplicity. Now I know that I was wrong.

The last major piece of data visualization that I was interested in doing was a pie chart that detailed an overall breakdown of how many hours were spent working on each section of the song. This was a relatively simple piece, as all I had to do was take the individual section totals that I had already calculated, add the estimated hours of previous work to their corresponding sections, and then apply that data to the chart itself, which can be seen below:



As we can see here, the first verse took the longest amount of time since it contained the 30 seconds of rotoscope animation on ones (24 frames per second = 720 frames, each frame took 10-15 minutes). In second place is the guitar solo, which makes sense because as mentioned before, most of the previous work was reworked/refitted here. New abstract work was added as well, such as the water drop and the three firework explosions. The remaining bulk of work was in the second verse, mostly due to the character animation (run cycles) as well as the waving English flag in the final shot. Finally, the intro/outro/compositing all collectively made up 84 hours (9.4%) of work.

In conclusion, this report was a good practice in breaking down different data points for visualization and analysis. To be able to apply it to a personal, real-world scenario was also fun since I knew that each days' numbers were accurate and the end result of a report would be a great piece to use in furthering my knowledge of data science. I also gained an ability to accurately estimate how long different animation pieces would take as I was working on them. There were some sections of the project where I would literally predict what day I would finish based on how many hours I felt were remaining, so that I could move onto the next section. Being able to plan around these numbers was an enormous step forward creatively as well, since I'll be able to manage future projects with even stronger accuracy by applying these concepts in gathering and analyzing data.